Python Fun(damentals)

Scaffolding your Python project

Alexander Rymdeko-Harvey

Obscurity Labs

```
* Directory Structure
* Config Files
```

Directory Structure

- With Python its critical to properly structure your code for the future.
- If done improperly it will inevitability cause headache and pain.
- Almost all Frameworks provide a "blueprint" or scaffolding utility.
    - A great example of this is Django.
    - A very opinionated structure that is used for building reusable web app projects.
- With out proper scaffolding publishing Python applications will be difficult and you will often run into issues with the build systems.

Defacto Python Structure

The following is the defacto Python module directory structure.

```
.
├── docs # project documentation
│   ├── conf.py
│   ├── index.rst
│   ├── make.bat
│   └── Makefile
├── LICENSE
├── Makefile
├── MANIFEST.in
├── README.rst
├── requirements.txt
├── sample # source code
│   ├── core.py
│   ├── helpers.py
│   └── __init__.py
├── setup.py # setuptools package
└── tests # unit and integration tests
    ├── context.py
    ├── __init__.py
    ├── test_advanced.py
    └── test_basic.py
```

Modern Python Webapp / CLI / Module

A progressive approach can be used with modern tooling and build systems.

```
awesome_toolset
├── Dockerfile <-- Used Testing - Publishing - Public Consumption
├── docs <-- Can be now be published with MkDocs etc.
│   └── index.md
├── Makefile
├── MANIFEST.in
├── mkdocs.yml
├── mypy.ini
├── poetry.lock
├── awesome_toolset <-- Named module or project
│   ├── __init__.py
│   └── main.py
├── pylint.ini
├── pytest.ini
├── README.md
├── scripts <-- Various dev scripts
│   └── configure_project.sh
├── setup.cfg
├── setup.py
└── tests <-- Store unit tests for Pytest etc.
    ├── __init__.py
    ├── conftest.py
    └── test_awesome_toolset.py
```

Config & Build System Files

Common files you may endup with:

- .pylintrc - PyLint
- .flake8 - Flake8 (PEP8 checker)
- .coveragerc - Coverage (Code coverage reporting)
- mypy.ini - Static Typing (Static analysis)
- pytest.ini - Pytest (Unit tests)

All of these tools are amazing and provide your project tremendous value. But they *can* clutter your workspace.

Combining Config & Build System Files

The new PEP 518 -- Specifying Minimum Build System Requirements for Python Projects. Brings us power to combine these files in some cases! Allowing modern build systems that follow PEP 518 to run test suites etc.

This can be added to Poetry `pyproject.toml` or `setup.cfg`:

```
[coverage:run]
branch = True
omit =
    */__main__.py
    ..SNIP..

[coverage:report]
exclude_lines =
    pragma: no cover
    ..SNIP..

[coverage:html]
directory = reports

[pylint]
...SNIP..
```